**ESTÁNDAR DE CODIFICACIÓN**

ASP .NET

UNIVERSIDAD VERACRUZANA

Facultad de Estadística e Informática

Tabla de contenido

[1 Introducción 2](#_Toc120521212)

[1.1 Propósito y alcance 2](#_Toc120521213)

[1.2 Idioma 2](#_Toc120521214)

[2 Convenciones de nombres 2](#_Toc120521215)

[2.1 Reglas comunes 2](#_Toc120521216)

[2.2 Reglas específicas 3](#_Toc120521217)

[2.2.1 Clases 3](#_Toc120521218)

[2.2.2 Paquetes 4](#_Toc120521219)

[2.2.3 Funciones 4](#_Toc120521220)

[2.2.4 Variables 4](#_Toc120521221)

[2.2.5 Constantes 4](#_Toc120521222)

[2.3 Prefijos permitidos 5](#_Toc120521223)

[3 Estilo código 5](#_Toc120521224)

[3.1 Espaciado 5](#_Toc120521225)

[3.2 Comentarios 7](#_Toc120521226)

[3.2.1 Mono línea 7](#_Toc120521227)

[3.2.2 Multilínea 7](#_Toc120521228)

[4 Buenas prácticas 7](#_Toc120521229)

[4.1 Manejo de errores y excepciones 7](#_Toc120521230)

[4.2 Recomendaciones 9](#_Toc120521231)

[5 Programación defensiva 10](#_Toc120521232)

[6 Referencias 14](#_Toc120521233)

# Introducción

En este documento se presentan las convenciones permitidas para la codificación de los proyectos ASP .NET que se desarrollan dentro de la Facultad de Estadística e Informática de la Universidad Veracruzana. Engloba las cuestiones de nombre permitidos en todos los componentes del sistema. Estilo de código y buenas prácticas

## Propósito y alcance

El presente documento se presentan las reglas, acuerdos y estándares de codificación que se seguirán para llevar a cabo los proyectos realizados en la Facultad de Estadística e Informática que sean desarrollados en ASP .NET. Esto con el propósito de desarrollar código de calidad, mejorar la legibilidad y hacer uso de las buenas prácticas existentes.

## Idioma

El idioma para desarrollar los proyectos ASP .NET, será inglés y español, como condicional que exista consistencia en el idioma seleccionado. Esta decisión, debido a que la gran mayoría de los proyectos a desarrollar en esta tecnología dentro de la FEI, son sistemas planeados para uso local o estatal, y solo unos pocos para un alcance nacional. Por lo que, el idioma no sería un limitante.

# Convenciones de nombres

A continuación, se describen las reglas que se aplicarán en el nombramiento de paquetes, clases, funciones, constantes y variables. Dicha sección, se divide en tres subsecciones, las cuales son las reglas comunes, reglas específicas y los prefijos permitidos en los proyectos a desarrollar.

## Reglas comunes

El presente apartado hace referencia a las convenciones que se aplicaran a todas las secciones mencionadas a continuación.

R-01: En los nombramientos largos, se omitirán las preposiciones.

Ejemplo incorrecto:

void takeOutOfArrayTheFirst**(**int **[]** arrayToChange**);**

Ejemplo correcto:

void takeOutArrayFirst**(**int **[]** arrayChange**);**

R-02: No se permitirá el uso de caracteres especiales en la declaración de nombres.

Ejemplo incorrecto:

int pair\_number**;**

Ejemplo correcto:

int pairNumber**;**

R-03: Los nombres deben ser descriptivos y de acuerdo con su propósito.

Ejemplo incorrecto:

int pn**;**

Ejemplo correcto:

int pairNumber**;**

## Reglas específicas

En este apartado, se presentan los acuerdos y los estilos de escritura a implementar según cada sección.

### Clases

R-01: Se usará la notación UpperCamelCase

Ejemplo incorrecto:

**public** class car**{** **}**

Ejemplo correcto:

**public** class Car**{** **}**

R-02: Los atributos y métodos de una clase deben estar en lowerCamelCase

Ejemplo incorrecto:

**public** class Person**{**

**private** int Age**;**

**public** int GetAge**(){**

**return** Age**;**

**}**

**}**

Ejemplo correcto:

**public** class Person**{**

**private** int age**;**

**public** int getAge**(){**

**return** age**;**

**}**

**}**

R-03: Los métodos de las clases, deben ser nombrados con un verbo en infinitivo

Ejemplo incorrecto:

void orderedArray**(**int **[]** array**);**

Ejemplo correcto:

void sort**(**int **[]** arreglo**);**

R-04: Las clases se deben nombrar con un sustantivo y en singular

Ejemplo incorrecto:

**public** class RedAirplanes**{** **}**

Ejemplo correcto:

**public** class Airplane**{** **}**

### Paquetes

R-01: La declaración de todos los paquetes, debe ser en minúsculas

Ejemplo incorrecto:

namespace Menu**;**

Ejemplo correcto:

namespace menu**;**

### Funciones

R-01: Para la declaración de funciones se usará la notación lowerCamelCase

Ejemplo incorrecto:

float CalculateAverage**(**int**[]** averageArray**);**

Ejemplo correcto:

float calculateAverage**(**int**[]** averageArray**);**

### Variables

R-01: Las variables deben ser declaradas con la notación lowerCamelCase

Ejemplo incorrecto:

float FinalAverage**;**

Ejemplo correcto:

float finalAverage**;**

### Constantes

R-01: Las constantes declaradas, deberán ser todo escrito en mayúsculas.

Ejemplo incorrecto:

int Pi **=** 3.1416**;**

Ejemplo correcto:

int PI **=** 3.1416**;**

R-02: Las constantes que cuenten con dos o más palabras, deberán ir separadas por un guion bajo

Ejemplo incorrecto:

int POSTALCODE **=** 1012**;**

Ejemplo correcto:

int POSTAL\_CODE **=** 1012**;**

## Prefijos permitidos

A continuación, se presenta una tabla donde se muestran los prefijos que se permitirán para los proyectos a realizar.

|  |  |
| --- | --- |
| Prefijo | Significado |
| btn | Button |
| lb | Label |
| tf | TextField |
| pn | Panel |
| ta | TextArea |
| cbx | CheckBox |
| rb | RadioButton |
| lt | List |
| cb | ComboBox |
| mb | MenuBar |
| tb | Table |
| ex | Excepciones |
| lv | ListView |
| dp | DatePicker |
| I | Interface |
| tv | TableView |
| tc | TableColumn |
| sp | ScrollPane |
| pf | PasswordField |
| tg | ToggleGroup |

# Estilo código

En esta sección, se abordarán los lineamientos para el estilo de código de los proyectos a desarrollar en ASP.NET

## Espaciado

Con el fin de proporcionar una mejor lectura de código, a continuación, se describen los lineamientos para el espaciado a utilizar en el código.

R-01: Posterior al cierre de una función, se deberá dejar una línea en blanco.

Ejemplo incorrecto:

**public** class Person**{**

**private** int age**;**

**public** int getAge**(){**

**return** age**;**

**}**

**public** void setAge**(**int age**){**

**this.**age **=** age**;**

**}**

**}**

Ejemplo correcto:

**public** class Person**{**

**private** int age**;**

**public** int getAge**(){**

**return** age**;**

**}**

**public** void setAge**(**int age**){**

**this.**age **=** age**;**

**}**

**}**

R-02: Luego de declarar los atributos de una clase, se deberá dejar una línea en blanco antes de la declaración de métodos.

Ejemplo incorrecto:

**public** class Person**{**

**private** int age**;**

**public** int getAge**(){**

**return** age**;**

**}**

**public** void setAge**(**int age**){**

**this.**age **=** age**;**

**}**

**}**

Ejemplo correcto:

**public** class Person**{**

**private** int age**;**

**public** int getAge**(){**

**return** age**;**

**}**

**public** void setAge**(**int age**){**

**this.**age **=** age**;**

**}**

**}**

## Comentarios

A continuación, se describen los acuerdos respecto a los comentarios, así como los casos donde se podrán utilizar.

### Mono línea

R-01: En los casos donde solo sea un comentario con una línea de código, estos no deberán superar los 45 caracteres.

Ejemplo incorrecto:

// Hello! There is my function “isPair”

Ejemplo correcto:

// Function by: < name >

### Multilínea

R-01: Los comentarios multilínea, se usarán para especificar si falta alguna implementación, ya sea de alguna clase o función.

Ejemplo incorrecto:

/\* Hello! There is my function “isPair”

The fuction check if a number is pair and return true. \*/

boolean isPair**(**int numberEvaluate**);**

Ejemplo correcto:

/\* Lack implement behavior of fuction

Just I wrote declaration to function \*/

boolean isPair**(**int numberEvaluate**);**

# Buenas prácticas

La presente sección, hace referencia a las buenas prácticas, las cuales conducirán a la elaboración de productos de calidad, así como recomendaciones para seguir el estándar.

## Manejo de errores y excepciones

A la hora de ejecución de un sistema, pueden surgir errores no previstos, por lo cual, a continuación, se describen como se describirán dichos errores.

R-01: Se debe especificar el error que haya sido generado por la excepción

Ejemplo incorrecto:

**try** **{**

int numberJoined**;**

numberJoined**=**Integer**.**parseInt**(**dataJoined**.**nextLine**());**

System**.out.**println**(**“Your number **is:** ” **+** numberJoined**);**

**}** **catch** **(**Exception exception**)** **{**

System**.out.**println**(**“ERROR”**);**

**}**

Ejemplo correcto:

**try** **{**

int numberJoined**;**

numberJoined**=**Integer**.**parseInt**(**dataJoined**.**nextLne**());**

System**.out.**println**(**“Your number **is:** ” **+** numberJoined**);**

**}** **catch** **(**NumberFormatException exNumeroInvalido**)** **{**

System**.out.**println**(**“Invalid number”**);**

**}** **finally** **{**

ReadNumber**();**

**}**

R-02: En las excepciones donde no se necesite interrumpir la ejecución de golpe, se deberá usar un bloque *finally.*

Ejemplo incorrecto:

**try** **{**

int numberJoined**;**

numberJoined**=**Integer**.**parseInt**(**dataJoined**.**nextLine**());**

System**.out.**println**(**“Your number **is:** ” **+** numberJoined**);**

**}** **catch** **(**Exception exception**)** **{**

System**.out.**println**(**“Invalid Number”**);**

**}**

Ejemplo correcto:

**try** **{**

int numberJoined**;**

numberJoined**=**Integer**.**parseInt**(**dataJoined**.**nextLne**());**

System**.out.**println**(**“Your number **is:** ” **+** numberJoined**);**

**}** **catch** **(**NumberFormatException exNumeroInvalido**)** **{**

System**.out.**println**(**“Invalid number”**);**

**}** **finally** **{**

// Block of code that runs

ReadNumber**();**

**}**

R-03: Las posibles excepciones que necesiten comunicarse en las capas superiores, se manejarán mediante el “encadenamiento de excepciones” creando una excepción personalizada.

Ejemplo incorrecto:

**try** **{**

resultSet**.**next**();**

int nombre **=** resultSet**.**getString**(**“nombre”**);**

**}** **catch** **(**SQLException ex**)** **{**

Log**.**logException**(**ex**);**

**}**

Ejemplo correcto:

If**(**resultSet**.**next**()** **==** **false){**

Throw **new** BusinessException**(**“academic **group** not found”**);**

**}**

## Recomendaciones

En esta sección se enumeran una serie de recomendaciones que son consideradas como buenas prácticas.

REC-01: Se debe procurar que las funciones hagan la menor cantidad de tareas posibles.

Ejemplo incorrecto:

float calculateAverageStudent**(**int**[]** grades**){**

float additionSubjects**=**0**;**

float totalSubjects**=**grades**.**length**();**

**for(**int i**=**0**;** i**<**Subjects**;** i**++){**

additionSubjects**=**additionSubjects**+**grades**[**i**];**

**}**

**for(**int i**=**0**;** i**<**totalSubjects**;** indice**++){**

System**.out.**println**(**“Grade**:** ”**+**grades**[**i**]);**

**}**

**return** additionSubjects**/**totalSubjects**;**

**}**

Ejemplo correcto:

float calculateAverage**(**float additionSubjects**,** float totalSubjects**){**

float average**=**0**;**

average **=** additionSubjects**/**totalSubjects**;**

**return** average**;**

**}**

float calculateAddition**(**int**[]** grades**){**

floatfloat additionSubjects**=**0**;**

int totalSubjects**=**grades**.**length**();**

**for(**int i**=**0**;** i**<**Subjects**;** i**++){**

additionSubjects**=**additionSubjects**+**grades**[**i**];**

**}**

**for(**int i**=**0**;** i**<**totalSubjects**;** indice**++){**

System**.out.**println**(**“Grade**:** ”**+**grades**[**i**]);**

**}**

**return** additionSubjects**;**

**}**

void showGrades**(**int**[]** grades**){**

**for(**int i**=**0**;** i**<**grades**.**length**();** i**++){**

System**.out.**println**(**“Grade**:** ”**+**grades**[**i**]);**

**}**

**}**

REC-02: Los atributos deben estar encapsulados

Ejemplo incorrecto:

**public** class Person**{**

**public** int age**;**

**}**

Ejemplo correcto:

**public** class Person**{**

**public** int age**;**

**}**

**public** class Person**{**

**private** int age**;**

**public** int getAge**(){**

**return** age**;**

**}**

**public** void setAge**(**int age**){**

**this.**age **=** age**;**

**}**

# Programación defensiva

A continuación, se presentan una serie de reglas relacionadas con la programación defensiva, esto con el fin de brindar mayor seguridad al código.

PD-01: Como primera regla, es que se debe minimizar el ámbito de las variables

Ejemplo incorrecto:

**public** class PorcentajesController **:** Controller

**{**

**private** var result **=** **null;**

**private** **readonly** ApplicationDbContext \_context**;**

**public** PorcentajesController**(**ApplicationDbContext context**)**

**{**

\_context **=** context**;**

**}**

**public** IActionResult Index**()**

**{**

result **=** JoinData**();**

**return** View**(**result**);**

**}**

Ejemplo correcto:

**public** class PorcentajesController **:** Controller

**{**

**private** **readonly** ApplicationDbContext \_context**;**

**public** PorcentajesController**(**ApplicationDbContext context**)**

**{**

\_context **=** context**;**

**}**

**public** IActionResult Index**()**

**{**

var result **=** JoinData**();**

**return** View**(**result**);**

**}**

PD-02: Se debe proveer retroalimentación del valor resultante de un método

Ejemplo incorrecto:

**public** boolean save**(**Student student**)** throws BusinessException**{**

boolean **value=false;**

**try** **{**

Connector connectorDataBase**=new** Connector**();**

Connection connectionDataBase **=** connectorDataBase**.**getConnection**();**

String insertStudent **=** "INSERT INTO Estudiante(matricula, “ + “nombre) VALUES (?,?)"**;**

PreparedStatement preparedStatement**;**

preparedStatement **=** connectionDataBase**.**prepareStatement**(**insertStudent**);**

preparedStatement**.**setString**(**1**,** student**.**getEnrollment**());**

preparedStatement**.**setString**(**2**,** student**.**getName**());**

preparedStatement**.**executeUpdate**();**

**value=true;**

connectorDataBase**.**disconnect**();**

**}** **catch** **(**SQLException sqlException**)** **{**

**throw** **new** BusinessException**(**"Connection failed"**,** sqlException**);**

**}** **catch** **(**ClassNotFoundException ex**)** **{**

Log**.**logException**(**ex**)**

**}**

**return** **value;**

**}**

Ejemplo correcto:

**public** List**<**PorcentajeCostos**>** ListaPorcentajes**()**

**{**

List**<**PorcentajeCostos**>** listaPorcentajes **=** **new** List**<**PorcentajeCostos**>();**

**using** **(**\_context**)**

**{**

var listado **=** **(from** pv **in** \_context**.**PuntoVenta

**join** pc **in** \_context**.**PorcentajeCostos on pv**.**Id equals pc**.**PuntoVentaId

**join** c **in** \_context**.**Clasificacion on pc**.**ClasificacionId equals c**.**Id

**select** **new**

**{**

puntoVentaName **=** pv**.**Nombre**,**

puntoVentaId **=** pv**.**Id**,**

clasificacionId **=** c**.**Id**,**

clasificacionName **=** c**.**Nombre**,**

insumos **=** pc**.**PorcentajeCostoInsumo**,**

variables **=** pc**.**PorcentajeCostoVariable**,**

fijos **=** pc**.**PorcentajeCostoFijo**,**

utilidad **=** pc**.**PorcentajeUtilidad

**}**

**);**

**foreach** **(**var item **in** listado**)**

**{**

PorcentajeCostos porcentajeCostos **=** **new** PorcentajeCostos**();**

PuntoVenta puntoVenta **=** **new** PuntoVenta**();**

Clasificacion clasificacion **=** **new** Clasificacion**();**

puntoVenta**.**Nombre **=** item**.**puntoVentaName**;**

clasificacion**.**Nombre **=** item**.**clasificacionName**;**

puntoVenta**.**Id **=** item**.**puntoVentaId**;**

clasificacion**.**Id **=** item**.**clasificacionId**;**

porcentajeCostos**.**PuntoVenta **=** puntoVenta**;**

porcentajeCostos**.**Clasificacion **=** clasificacion**;**

porcentajeCostos**.**PorcentajeCostoInsumo **=** item**.**insumos**;**

porcentajeCostos**.**PorcentajeCostoVariable **=** item**.**insumos**;**

porcentajeCostos**.**PorcentajeCostoFijo **=** item**.**fijos**;**

porcentajeCostos**.**PorcentajeUtilidad **=** item**.**utilidad**;**

listaPorcentajes**.**Add**(**porcentajeCostos**);**

**}**

**}**

**return** listaPorcentajes**;**

**}**

PD-03: Se debe habilitar la verificación de tipos en tiempo de compilación

Ejemplo incorrecto:

**public** List**<>** getLGACs**(){**

**return** lgacs**;**

**}**

**public** void setLGACs**(**List**<>** lgacs**){**

**this.**lgacs**=**lgacs**;**

**}**

Ejemplo correcto:

**public** List**<**LGAC**>** getLGACs**(){**

**return** lgacs**;**

**}**

**public** void setLGACs**(**List**<**LGAC**>** lgacs**){**

**this.**lgacs**=**lgacs**;**

**}**

PD-05: Seguridad de la base de datos

Ejemplo incorrecto:

**private** Connection connection**;**

**private** String host **=** "localhost"**;**

**private** int port **=** 3306**;**

**private** String dataBaseName **=** "CuerpoAcademico"**;**

**private** String url **=** "jdbc:mysql://" **+** host **+** ":" **+** String**.**valueOf**(**port**)** **+** "/" **+** dataBaseName**;**

**private** String userName **=** "root"**;**

**private** String userPassword **=** "password"**;**

Ejemplo correcto:

**{**

"ConnectionStrings"**:** **{**

"DefaultConnection"**:** "Server=localhost;Database=SanRafael;Trusted\_Connection=True;MultipleActiveResultSets=true"**,**

**},** //localhost;

"Logging"**:** **{**

"LogLevel"**:** **{**

"Default"**:** "Warning"**,**

"Microsoft.\*"**:** "Warning"

**}**

**}**

**}**

PD-06: Verificación de entradas

Ejemplo incorrecto:

GroupAcademic groupAcademicAuxiliar **=new** GroupAcademic**(**key**,**name**,**objetive**,**consolidationGrade**,**mision**,**vision**);**

GroupAcademicDAO groupAcademicDAO **=new** GroupAcademicDAO**();**

groupAcademicDAO**.**savedSucessful**(**groupAcademicAuxiliar**);**

recoverLgcas**();**

AlertMessage alertMessage**=new** AlertMessage**();**

alertMessage**.**showMessageSave**(**"Cuerpo Academico"**);**

Ejemplo correcto:

**public** class PorcentajeCostos

**{**

**public** int**?** ClasificacionId **{** **get;** **set;** **}**

**public** int**?** PuntoVentaId **{** **get;** **set;** **}**

**[**ForeignKey**(**"ClasificacionId"**)]**

**public** Clasificacion Clasificacion **{** **get;** **set;** **}**

**[**ForeignKey**(**"PuntoVentaId"**)]**

**public** PuntoVenta PuntoVenta **{** **get;** **set;** **}**

**[**Display**(**Name **=** "Porcentaje Insumos"**)]**

**[**Required**(**ErrorMessage **=** "El campo {0} es requerido."**)]**

**public** decimal PorcentajeCostoInsumo **{** **get;** **set;** **}**

**[**Display**(**Name **=** "Porcentaje costo de variable"**)]**

**[**Required**(**ErrorMessage **=** "El campo {0} es requerido."**)]**

**public** decimal PorcentajeCostoVariable **{** **get;** **set;** **}**

**[**Display**(**Name **=** "Porcentaje de costo fijo"**)]**

**[**Required**(**ErrorMessage **=** "El campo {0} es requerido."**)]**

**public** decimal PorcentajeCostoFijo **{** **get;** **set;** **}**

**[**Display**(**Name **=** "Porcentaje de utilidad"**)]**

**[**Required**(**ErrorMessage **=** "El campo {0} es requerido."**)]**

**public** decimal PorcentajeUtilidad **{** **get;** **set;** **}**

**}**

**}**

# Referencias

Microsoft .NET (2022). C# Coding Conventions. <https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions>